8WEEKSQLCHALLENGE.COM

CASE STUDY #1

THE TASTE OF SUCCESS

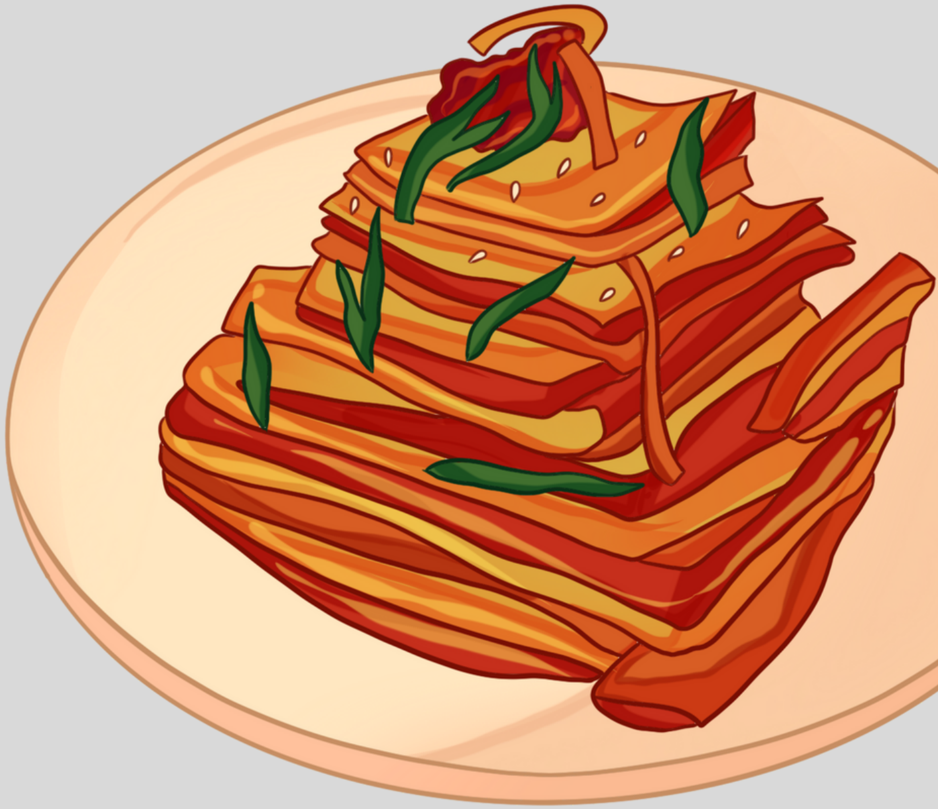
DATAWITHDANNY.COM

Created by :
SHRADDHA SHUKLA

INTRODUCTION

Danny seriously loves Japanese food so in the beginning of 2021, he decides to embark upon a risky venture and opens up a cute little restaurant that sells his 3 favourite foods: sushi, curry and ramen.

Danny’s Diner is in need of your assistance to help the restaurant stay afloat - the restaurant has captured some very basic data from their few months of operation but have no idea how to use their data to help them run the business.



PROBLEM STATEMENT

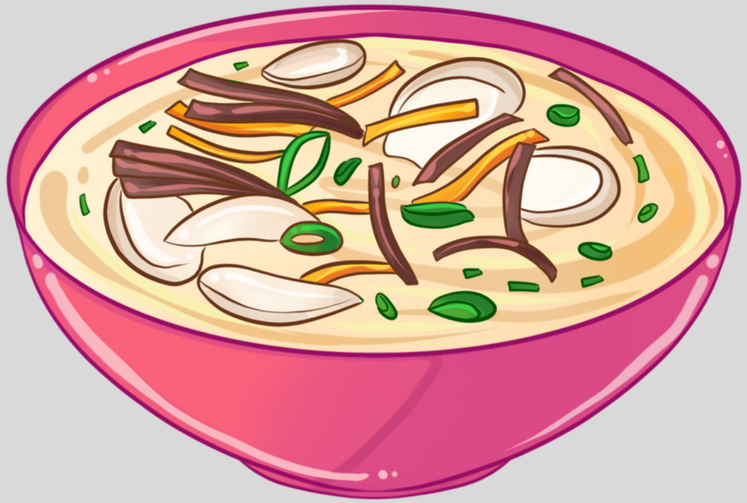
Danny wants to use the data to answer a few simple questions about his customers, especially about their visiting patterns, how much money they've spent and also which menu items are their favourite. Having this deeper connection with his customers will help him deliver a better and more personalised experience for his loyal customers.

He plans on using these insights to help him decide whether he should expand the existing customer loyalty program - additionally he needs help to generate some basic datasets so his team can easily inspect the data without needing to use SQL.

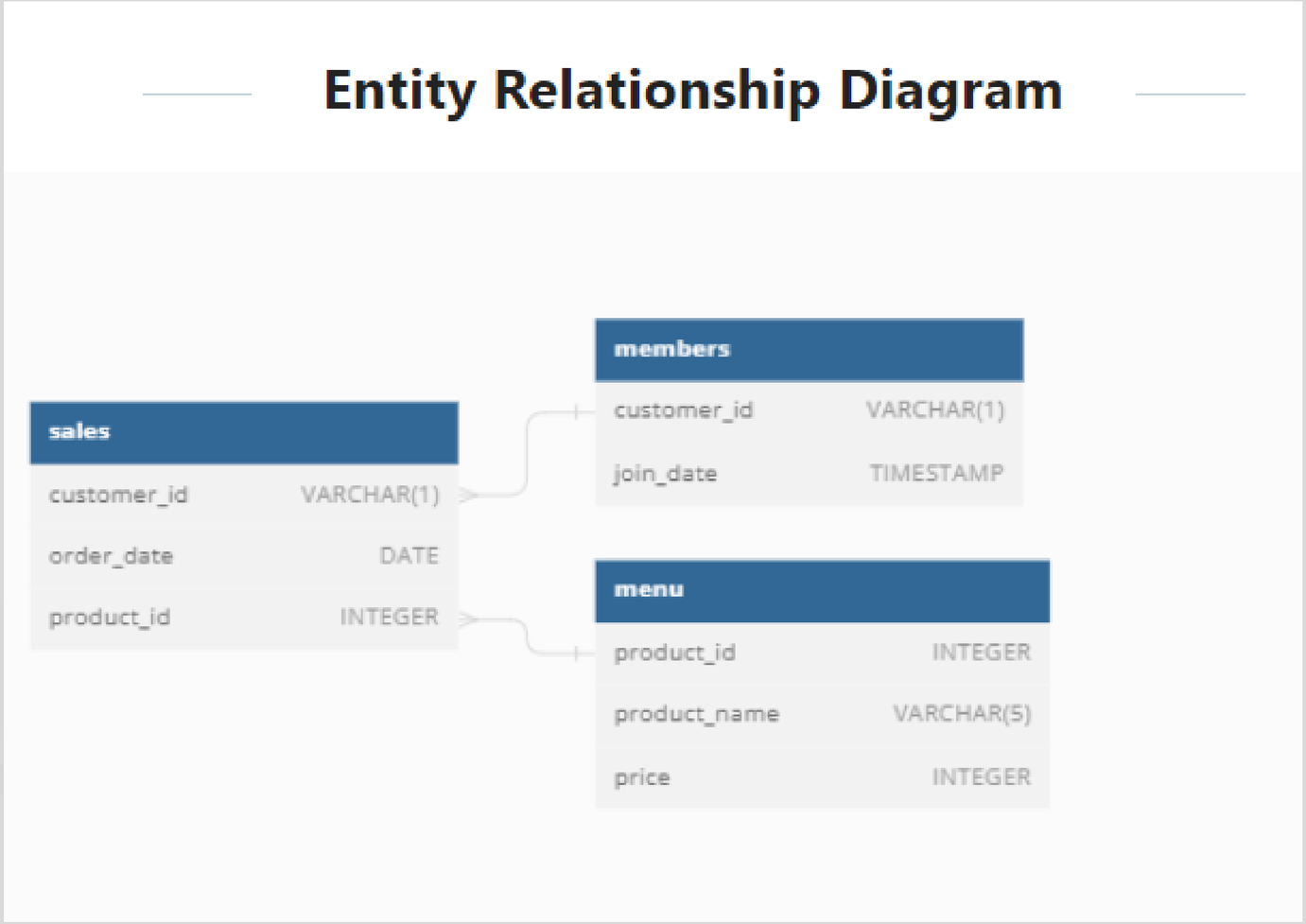
Danny has provided you with a sample of his overall customer data due to privacy issues - but he hopes that these examples are enough for you to write fully functioning SQL queries to help him answer his questions!

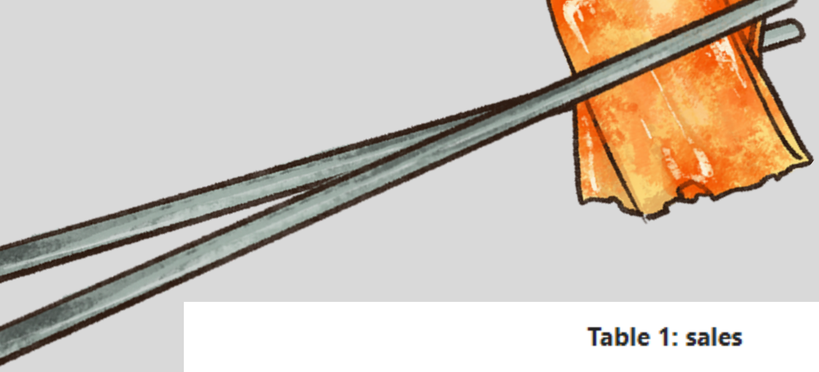
<u>Danny has shared with you 3 key datasets for this case study:</u>

sales
menu
members



ENTITY RELATIONSHIP DIAGRAM





TABLES

Table 1: sales

The `sales` table captures all `customer_id` level purchases with an corresponding `order_date` and `product_id` information for when and what menu items were ordered.

customer_id	order_date	product_id
A	2021-01-01	1
A	2021-01-01	2
A	2021-01-07	2
A	2021-01-10	3
A	2021-01-11	3
A	2021-01-11	3
B	2021-01-01	2
B	2021-01-02	2
B	2021-01-04	1
B	2021-01-11	1
B	2021-01-16	3
B	2021-02-01	3
C	2021-01-01	3
C	2021-01-01	3
C	2021-01-07	3

Table 2: menu

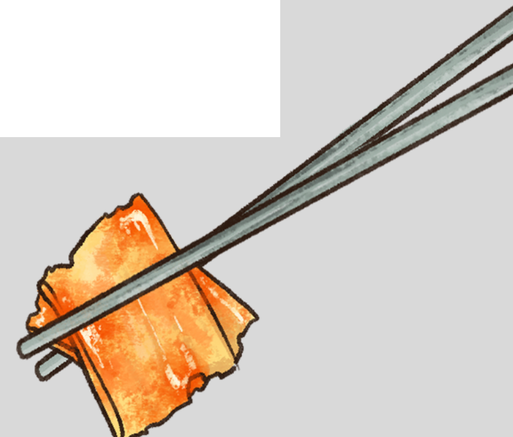
The `menu` table maps the `product_id` to the actual `product_name` and `price` of each menu item.

product_id	product_name	price
1	sushi	10
2	curry	15
3	ramen	12

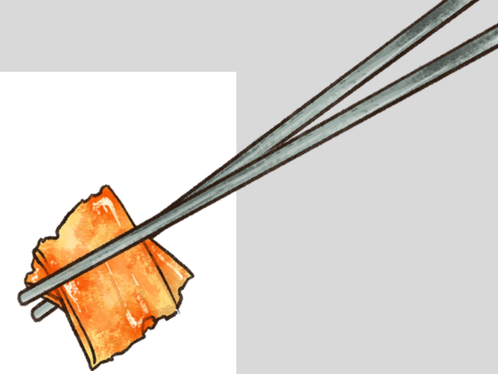
Table 3: members

The final `members` table captures the `join_date` when a `customer_id` joined the beta version of the Danny's Diner loyalty program.

customer_id	join_date
A	2021-01-07
B	2021-01-09



Case Study Questions



Each of the following case study questions can be answered using a single SQL statement:

1. What is the total amount each customer spent at the restaurant?
2. How many days has each customer visited the restaurant?
3. What was the first item from the menu purchased by each customer?
4. What is the most purchased item on the menu and how many times was it purchased by all customers?
5. Which item was the most popular for each customer?
6. Which item was purchased first by the customer after they became a member?
7. Which item was purchased just before the customer became a member?
8. What is the total items and amount spent for each member before they became a member?
9. If each $1 spent equates to 10 points and sushi has a 2x points multiplier - how many points would each customer have?
10. In the first week after a customer joins the program (including their join date) they earn 2x points on all items, not just sushi - how many points do customer A and B have at the end of January?

QUESTION 1:

What is the total amount each customer spent at the restaurant?

Query 1 | members | menu | sales | SQL File 3* | danny dinner...*

Limit to 500 rows

```
1
2    #1 What is the total amount each customer spent at the restaurant?
3
4    SELECT
5     s.customer_id,sum(m.price) AS total_amount_spent
6    FROM sales s JOIN menu m ON s.product_id=m.product_id
7    GROUP BY s.customer_id;
8
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	total_amount_spent
A	76
B	74
C	36

QUESTION 2:

How many days has each customer visited the restaurant?

Query 1 | members | menu | sales | SQL File 3* | danny dinner...*

Limit to 500 rows

```
10
11      #2. How many days has each customer visited the restaurant?
12
13      SELECT
14       customer_id, COUNT(DISTINCT order_date) number_of_days_customer_visited
15      FROM sales GROUP BY customer_id;
16
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	number_of_days_customer_visited
A	4
B	6
C	2

QUESTION 3:

What was the first item from the menu purchased by each customer?

Query 1 | members | menu | sales | SQL File 3* | danny dinner...*

Limit to 500 rows

```
16
17      #3. What was the first item from the menu purchased by each customer?
18
19      WITH cte AS
20      (
21      SELECT customer_id, order_date , product_id ,
22      ROW_NUMBER () OVER(PARTITION BY customer_id ORDER BY order_date) AS rn
23      FROM sales)
24      SELECT cte.customer_id ,cte.product_id, product_name
25      FROM cte
26      JOIN menu m  ON cte.product_id = m.product_id
27      WHERE rn = 1;
28
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	product_id	product_name
A	1	sushi
B	2	curry
C	3	ramen

QUESTION 4:

What is the most purchased item on the menu and how many times was it purchased by all customers?

```
28
29      #4. What is the most purchased item on the menu and how many times was it purchased by all customers?
30
31  •   WITH cte AS
32      (
33      SELECT product_id , COUNT(1) as purchased_frequency FROM sales GROUP BY product_id
34      )
35      SELECT m.product_name , cte.purchased_frequency FROM cte JOIN menu m
36      ON m.product_id=cte.product_id
37      ORDER BY purchased_frequency DESC
38      LIMIT 1 ;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

product_name	purchased_frequency
ramen	8

QUESTION 5:

Which item was the most popular for each customer?

```sql
#5. Which item was the most popular for each customer?

WITH cte AS
(SELECT
 s.customer_id , m.product_name FROM sales s
JOIN menu m ON s.product_id = m.product_id
) ,
cte2 AS
(
SELECT
  cte.customer_id , cte.product_name, count(cte.product_name) AS product_frequency,
DENSE_RANK() OVER(PARTITION BY cte.customer_id ORDER BY count(cte.product_name)DESC ) AS dnk
FROM cte
GROUP BY cte.customer_id , cte.product_name)
SELECT cte2.customer_id ,  cte2.product_name, cte2.product_frequency
FROM cte2 WHERE cte2.dnk = 1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	product_name	product_frequency
A	ramen	3
B	curry	2
B	sushi	2
B	ramen	2
C	ramen	3

QUESTION 6:

Which item was purchased first by the customer after they became a member?

Query 1 | members | menu | sales | SQL File 3* | danny dinner...*

Limit to 500 rows

```
58  #6. Which item was purchased first by the customer after they became a member?
59
60  WITH cte AS (
61  SELECT ms.customer_id , ms.join_date , mu.product_name ,s.product_id ,s.order_date,
62  ROW_NUMBER() OVER(PARTITION BY customer_id ORDER BY order_date) AS row_num
63  FROM members ms
64  JOIN sales s ON ms.customer_id = s.customer_id
65  JOIN menu mu ON mu.product_id= s.product_id
66  WHERE order_date >= join_date
67  )
68  SELECT cte.customer_id , cte.join_date ,
69  cte.product_name , cte.product_id ,cte.order_date FROM cte
70  WHERE row_num =1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	join_date	product_name	product_id	order_date
A	2021-01-07	curry	2	2021-01-07
B	2021-01-09	sushi	1	2021-01-11

QUESTION 7:

Which item was purchased just before the customer became a member?

Query 1 | members | menu | sales | SQL File 3* | danny dinner...*

Limit to 500 rows

```
73
74      #7. Which item was purchased just before the customer became a member?
75
76      WITH cte AS (
77      SELECT ms.customer_id , ms.join_date , mu.product_name , s.product_id ,s.order_date,
78      RANK() OVER(PARTITION BY customer_id ORDER BY order_date DESC) AS rnk
79      FROM members ms
80      JOIN sales s ON ms.customer_id = s.customer_id
81      JOIN menu mu ON mu.product_id= s.product_id
82      WHERE order_date < join_date
83      )
84      SELECT cte.customer_id , cte.join_date ,
85      cte.product_name , cte.product_id ,cte.order_date FROM cte
86      WHERE rnk =1;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	join_date	product_name	product_id	order_date
A	2021-01-07	sushi	1	2021-01-01
A	2021-01-07	curry	2	2021-01-01
B	2021-01-09	sushi	1	2021-01-04

QUESTION 8:

What is the total items and amount spent for each member before they became a member?

Query 1 | members | menu | sales | SQL File 3* | danny dinner...*

Limit to 500 rows

```
88
89      #8. What is the total items and amount spent for each member before they became a member?
90
91      WITH cte AS (
92      SELECT ms.customer_id , ms.join_date , mu.product_name , s.product_id ,s.order_date,mu.price
93       FROM members ms
94      JOIN sales s ON ms.customer_id = s.customer_id
95      JOIN menu mu ON mu.product_id= s.product_id
96      WHERE order_date < join_date
97      )
98      SELECT cte.customer_id , SUM(cte.price) AS amount_spent ,
99      COUNT(cte.product_name ) AS total_items FROM cte
100     GROUP BY  cte.customer_id ;
101
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	amount_spent	total_items
B	40	3
A	25	2

QUESTION 9:

If each $1 spent equates to 10 points and sushi has a 2x points multiplier - how many points would each customer have?

Query 1 | members | menu | sales | SQL File 3* | danny dinner...*

Limit to 500 rows

```
103
104     #9. If each $1 spent equates to 10 points and sushi has a 2x points multiplier -
105     #how many points would each customer have?
106
107 •   WITH cte AS (
108     SELECT s.customer_id,s.product_id,m.product_name,m.price,
109     CASE
110          WHEN m.product_name = "sushi" THEN price*20
111          WHEN m.product_name <> "sushi" THEN price*10
112          END AS points
113          FROM sales s
114          JOIN menu m ON s.product_id=m.product_id)
115     SELECT cte.customer_id, sum(points) AS total_points FROM cte GROUP BY cte.customer_id;
116
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	total_points
A	860
B	940
C	360

QUESTION 10:

In the first week after a customer joins the program (including their join date) they earn 2x points on all items, not just sushi - how many points do customer A and B have at the end of January?

Query 1 | members | menu | sales | SQL File 3* | danny dinner...*

Limit to 500 rows

```
118
119     #10. In the first week after a customer joins the program (including their join date) they earn
120     #2x points on all items, not just sushi - how many points do customer A and B have at the end of January?
121
122     WITH cte AS (
123     SELECT s.customer_id , s.order_date , s.product_id , m.join_date, mu.price*20 AS price_
124     FROM sales s JOIN members m  ON s.customer_id=m.customer_id
125     JOIN menu mu ON s.product_id = mu.product_id
126     WHERE m.join_date<=s.order_date AND MONTH(order_date)=1
127     )
128     SELECT cte.customer_id , SUM(cte.price_) FROM cte
129     GROUP BY cte.customer_id;
130
131
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	SUM(cte.price_)
B	440
A	1020

Bonus_Question-Join all the things

Query 1 | members | menu | sales | SQL File 3* | danny dinner...*

Limit to 500 rows

```
134
135 •  SELECT s.customer_id, s.order_date ,mu.product_name , mu.price ,
136    CASE
137    WHEN ms.join_date>s.order_date then "N"
138    ELSE "Y"
139    END AS member
140    FROM sales s JOIN menu mu ON s.product_id = mu.product_id
141    JOIN members ms on ms.customer_id = s.customer_id
142    ORDER BY s.customer_id,s.order_date;
143
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_id	order_date	product_name	price	member
A	2021-01-01	sushi	10	N
A	2021-01-01	curry	15	N
A	2021-01-07	curry	15	Y
A	2021-01-10	ramen	12	Y
A	2021-01-11	ramen	12	Y
A	2021-01-11	ramen	12	Y
B	2021-01-01	curry	15	N
B	2021-01-02	curry	15	N
B	2021-01-04	sushi	10	N
B	2021-01-11	sushi	10	Y
B	2021-01-16	ramen	12	Y
B	2021-02-01	ramen	12	Y

Bonus_Question- Rank all the things

Query 1 | members | menu | sales | SQL File 3* | danny dinner...*

Limit to 500 rows

```
146 •  WITH cte AS
147    (
148    SELECT s.customer_id, s.order_date ,mu.product_name , mu.price ,
149    CASE
150     WHEN ms.join_date>s.order_date THEN "N"
151     ELSE "Y"
152    END AS member
153    FROM sales s JOIN menu mu ON s.product_id = mu.product_id
154    JOIN members ms on ms.customer_id = s.customer_id
155    ORDER BY s.customer_id,s.order_date)
156    SELECT *,
157    CASE
158     WHEN cte.member = "N" THEN null
159    ELSE RANK() OVER (PARTITION BY cte.customer_id ,cte.member ORDER BY cte.order_date ASC )
160    END AS ranking
161    FROM cte;
```

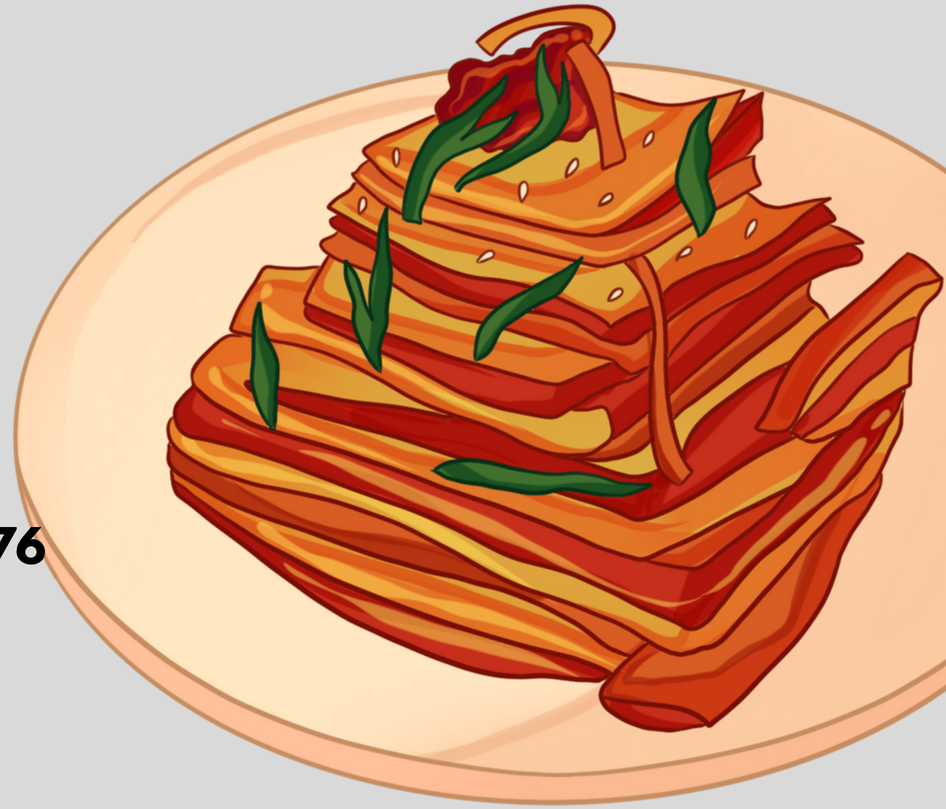
customer_id	order_date	product_name	price	member	ranking
A	2021-01-01	sushi	10	N	NULL
A	2021-01-01	curry	15	N	NULL
A	2021-01-07	curry	15	Y	1
A	2021-01-10	ramen	12	Y	2
A	2021-01-11	ramen	12	Y	3
A	2021-01-11	ramen	12	Y	3
B	2021-01-01	curry	15	N	NULL
B	2021-01-02	curry	15	N	NULL
B	2021-01-04	sushi	10	N	NULL
B	2021-01-11	sushi	10	Y	1
B	2021-01-16	ramen	12	Y	2
B	2021-02-01	ramen	12	Y	3

KEY INSIGHTS

Case study # 1 -The taste of success

Function used in this case study are :

- CTE(common table Expression),
- Group by , Aggregate function ,
- Joins,
- Window function - ranking , row_number,
- Case when clause



Insights gathered from case study are:

- Customer A spent maximum amount iin the restaurant i.e, $76 followed by B $74 and C $36.
- Customer B is one of the most frequent customer .
- Ramen is the most purchased item on the menu.
- Customer B has the maximum points i.e., 940 points in total, followed by customer A having 860 and Customer C having 360 points

THANK YOU